CHEAPS: a Checker of Asynchronous Parameterized Systems

Igor V. Konnov

Computer Systems Laboratory
Faculty of Computational Mathematics and Cybernetics
Lomonosov Moscow State University

The Third Workshop on Invariant Generation
WING 2010, Edinburgh
Outline

Parameterized Model Checking Problem

Invariant Generation

Tool Demonstration

Properties of Simulations (opt.)

Summary
Parameterized Model Checking Problem

- Given a temporal property $\varphi$ and an infinite family of distributed systems $\mathcal{F} = \{M_n\}$ composed of similar processes check $M_n \models \varphi$ for all the finite models from $\mathcal{F}$.
- In general, PMCP is undecidable [Apt, Kozen 86].
- Various cases of PMCP depend on:
  - communication topology of the family $\mathcal{F}$;
  - parallelism: synchronous, asynchronous;
  - synchronization primitives;
  - temporal properties: local (single-index), global (multi-index).
Network Grammars

- A family $\mathcal{F}$ is specified by a network grammar $G = (\mathcal{N}, \mathcal{T}, \mathcal{P}, S)$, i.e. models are induced by $G$ [Shtadler, Grumberg 90].
- Each symbol from $\mathcal{N} \cup \mathcal{T}$ induces the language of finite labelled transition systems.
- The start non-terminal $S$ induces the set $\mathcal{F} = \{M_n\}$ of finite LTSes.
- Production rules $N \rightarrow X_1[r_1] \parallel \cdots \parallel X_k[r_k]$ describe composition of LTSes derived from $X_i$ with action renaming $r_i$, $1 \leq i \leq k$. 

Igor V. Konnov (MSU) Checker of Asynchronous Parameterized Systems WING @ FLoC 2010 4 / 23
Network Invariants

- For each non-terminal $N$ find an invariant $I_N$ such that $\alpha(M_N) \preceq_{\text{sim}} \alpha(I_N)$ holds for:
  - all LTSes derived from $N$ in $G$,
  - an abstraction $\alpha$,
  - a simulation relation $\preceq_{\text{sim}}$.

- A property $\varphi$ is expressed using temporal logic over regular expressions [Clarke, Grumberg, Jha 95].

- By induction if invariants are found for all the non-terminals it suffices to check the property $\varphi$ on $\alpha(I_S)$. 
Abstraction is optional.

Asynchronous parallelism (as more realistic) is used.

Weaker simulations $\preceq$ for invariant detection are introduced: quasiblock, block, and semiblock. [Konnov, Zakharov 07]

Local properties ($\text{ACTL}^*_\chi$ formulae) of distinguished processes are checked.
Invariant Generation

Tool Motivation

- Constructing invariants of parallel processes manually is a very hard task.
- Invariants must be constructed automatically.
- Translation of existing model in (another) more expressive language is non-trivial and error-prone.
- It is desirable to use models described in a widespread language (PROMELA) and thus integrate our tool with well-known model checker (SPIN).
Invariant Generation

Tool Architecture

Prototypes (TinyPromela) - Network Grammar

Model Generator - Commander - Simulation Checker

Models (TinyPromela) - Invariant models

Properties (LTL) - Spin

Failpath

Traces
Invariant Generation

Modules

- **Model Generator** creates Promela models using a given network grammar: for a given parse tree, for a number of terminals, for a height of a parse tree.

- **Simulation Checker** constructs semiblock simulation between two models and checks that initial pairs belong to it.

- **Failpath** is a trace finder and a visualization tool. It helps a user to understand why Simulation Checker has failed to compute a desired simulation.
**Commander** module organizes enumeration procedure:

- for each non-terminal $N$ of $G$ it incrementally constructs candidate invariants $I_N$,
- for each candidate invariant $I_N$ of height $h$ it constructs LTSes $M_1, \ldots, M_k$ of height $h + 1$ induced from $N$;
- and checks that $M_i \preceq_{sbsim} I_N$ for $i : 1 \leq i \leq k$. 
Example: Awerbuch’s Distributed DFS

- Processes exchange by messages \texttt{tok}, \texttt{vis}, and \texttt{ack}.
- One distinguished process, Initiator, obtains \texttt{tok} first.
- Upon receipt of \texttt{tok} each node informs the neighbors by sending \texttt{vis} and waits for \texttt{ack} to be sent back.
- Finally, Initiator \textit{decides} when all its neighbours have informed it.
Example (cont.)
Example (cont.)
Example (cont.)

5::tok

i

ℓ

n

ℓ

ℓ
Example (cont.)
Example (cont.)
Example (cont.)
Example (cont.)

![Diagram of a tree structure with nodes labeled as follows:
- The root node is labeled $i$.
- There are two child nodes below $i$: one labeled $n$ and another labeled $l$.
- An edge labeled "13:tok" connects the root $i$ to the node $n$.
- An edge labeled "14:tok" connects the root $i$ to the node $l$.
- The node $n$ has two children: one labeled $l$ and another labeled $l$, with an edge labeled "14:tok" between them.
]
Example (cont.)

\[ 15:tok \]
Example (cont.)

decide!

\[
\begin{tikzpicture}
  \node {i} [circle,fill=blue!20,inner sep=1pt] (i) at (0,0) [] {};
  \node {n} [circle,fill=blue!20,inner sep=1pt] (n) at (1.5,0) [] {};
  \node {l} [circle,fill=blue!20,inner sep=1pt] (l1) at (1,0.5) [] {};
  \node {l} [circle,fill=blue!20,inner sep=1pt] (l2) at (1,-0.5) [] {};
  \draw (i) -- (l1);
  \draw (i) -- (l2);
  \draw (n) -- (l1);
  \draw (n) -- (l2);
  \draw (1.5,0.5) node [above] {16:tok};
\end{tikzpicture}
\]
Run the Demo
Explaining the Demo

processes {
    r[left, right] = Initiator2;
    n[parent, left, right] = Node3;
    l[parent] = Node1;
}
nonterminals {
    S[];
    N[parent, left, right];
}
rules {
    S => N[l/parent]
    || N[r/parent]
    || r[l/left, r/right];
    N => N[l/parent]
    || N[r/parent]
    || n[l/left, r/right];
    N => l[];
}

derived from N (height 3)
candidate invariant (height 2)
Explaining the Demo (cont.)
Other Parameterized Models

- Chandy-Lamport snapshot algorithm.
- Resource ReserVation Protocol (RSVP).
- Milner’s Scheduler (proven to have block bisimulation between rings [Emerson, Namjoshi 95], but there is no invariant for non-terminal).
Simulations

Semiblock simulation

Computed by simulation checker
Properties of Simulations (opt.)

Simulations

Semiblock simulation iff Block simulation

Computed by simulation checker

Preserves $\text{ACTL}^{* \chi}$

Not monotonic
Properties of Simulations (opt.)

Simulations

Semiblock simulation

 iff

Block simulation

 is a

Quasiblock simulation

Preserves $\text{ACTL}^*_{-X}$

Not monotonic

Preserves $\text{ACTL}^*_{-X}$

Monotonic

Computed by simulation checker
CHEAPS provides a procedure to find invariant models of parameterized family induced by a network grammar.

Invariant models are constructed and checked automatically.

Models written in a subset of PROMELA are given on input. This allows to integrate with SPIN easily.

The tool is available at my homepage:
http://lvk.cs.msu.su/~konnov
Thank you!

Questions?
Modes of Simulation Checker

- DFA state space representation from SPIN compresses the sets greatly \([-\text{dfa}]\).
- DFA + File representation representation allows us to speed up iterations over the sets \([-\text{dfafile}]\).
- Partitioning of the relation into stable and unstable subsets allows us to avoid redundant checks (default).
- Back propagation of negative results decreases the number of iterations and states to be checked \([-\text{back}]\).
- Partial order reduction decreases the time of front construction \([-\text{optbld}]\).
Failpath

- In the case invariants are not found...
- Demo..?
Semiblock simulation

For any pair of states \((s_1, t_1) \in H\) the following conditions hold:

- \(L_1(s_1) \cap \Sigma_0 = L_2(t_1) \cap \Sigma_0\).
- For any finite block from \(s_1\) there exists a finite block from \(t_1\) such that \((s_{m+1}, t_{n+1}) \in H\) and \(n > 1\) implies \((s_1, t_n) \in H\).
- For any infinite block from \(s_1\) there exists an infinite block from \(t_1\) and \(k \in \mathbb{N}\) such that \((s_1, t'_k) \in H\).
E.M. Clarke, O. Grumberg, S. Jha.  
Verifying parameterized networks using abstraction and regular languages.  

I. V. Konnov, V. A. Zakharov.  
An invariant-based approach to the verification of asynchronous parameterized networks.  
Accepted for publication in Journal of Symbolic Computation, 2009.